Web Access to the DZero Significant Events System

Judith Odili

Lincoln University

SIST Program

Fermi National Laboratory

Batavia, IL

August 11, 2005

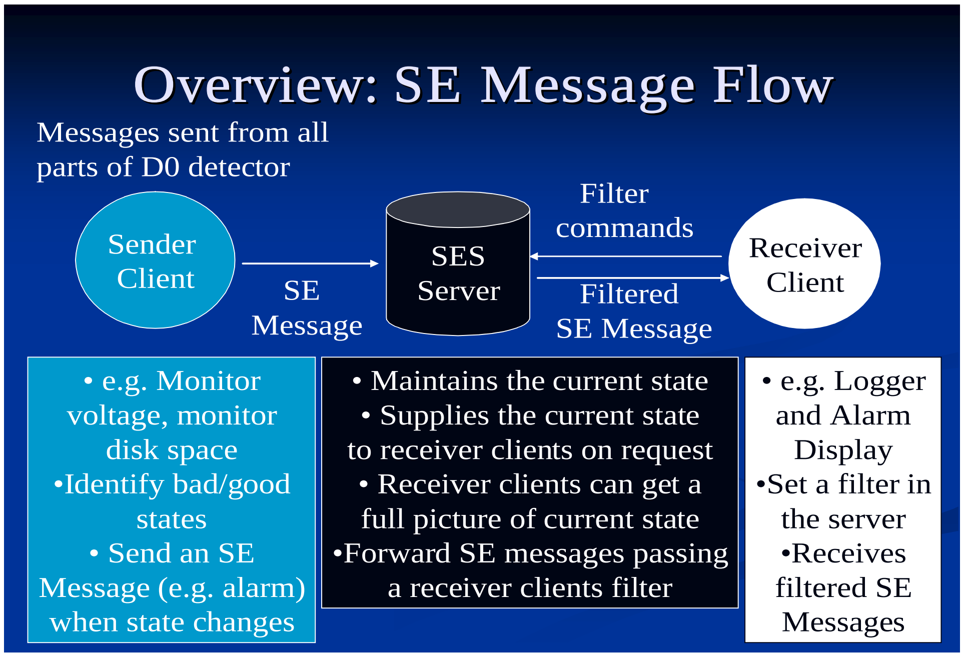
Research Supervisor: Geoff Savage

INTRODUCTION

Fermi National Accelerator Laboratory advances the understanding of the fundamental nature of matter and energy by providing leadership resources for qualified researchers to conduct basic research at the frontiers of high energy physics and related disciplines. Fermi lab has a two-mile Main injector accelerator that increases the number of proton-antiproton collisions in the Tevatron, greatly enhancing chances for important discoveries in Run II. Physicists need more than accelerators in order to "see" high energy collisions. That is why scientists have designed and built particle detectors, huge "cameras" that can take more than a million "snapshots" of particle collisions every second. At Fermilab, two huge detectors called CDF and D-Zero, both consisting of many different detection subsystems, are located in the Tevatron beam line. The particle collisions take place at the centers of these collider detectors. The detectors observe the collisions, recognize the particles that come "flying" out and record all information for later analysis.

THE SIGNIFICANT EVENT SYSTEM

The Significant Event System (SES) has been developed principally for the recording and reporting of events detected by the D-Zero detector. The alarm SES has the capability of handling 'alarms', events, run controls, and other event notifications. Alarms are used to report the occurrence of an event. They may either report a stage change (good to bad, or vice versa) or report information. All the significant event messages are sent to a "Central Server", and this server maintains the current state, and supplies the current state to receiver clients. Usually, the receiver clients send a filter to the server and the messages are forwarded to the receiver client based on the filter parameters specified.


Overview: SE Message Flow
Messages sent from all parts of D0 detector
Sender Client
SE Message
SES Server
Filter commands
Filtered SE Message
Receiver Client
• e.g. Monitor voltage, monitor disk space
•Identify bad/good states
• Send an SE Message (e.g. alarm) when state changes
• Maintains the current state
• Supplies the current state to receiver clients on request
• Receiver clients can get a full picture of current state
•Forward SE messages passing a receiver clients filter
• e.g. Logger and Alarm Display
•Set a filter in the server
•Receives filtered SE Messages

THE SIGNIFICANT EVENTS MESSAGE FLOW

The logger is one of the main receiver clients. It receives all the SE messages from the server, and writes it to a hard disk in form of Log files. A new Log file is created everyday, and a new log file is also created whenever the logger is restarted. To access any of the messages written in these log files, a python program called the "SesBrowser.py" has been written. This program was created to enable specific messages of interest to be read from the Log files. It reads filter parameters from a configuration file, as well as the time range of the messages required, and it displays the results on the screen, according to the specified display format. The configuration file contains the filter parameters, including the detector, and device of interest. It also includes the 'id' of the message and the time range of the required messages.

A simple configuration file would look like this:

```
addFilter(id='HV',filter='contains(name,"MUO")',action='print()')
setTime(start='2005-06-20 16:00:00',end='2005-06-20 16:05:00')
```

Of course, it can get a lot more complicated than that. This sample configuration file (assuming the name browser.fil) has the time period specified, so the command would look something like this:

```
> setup d0online
 > sesBrowser.py browser.fil
```

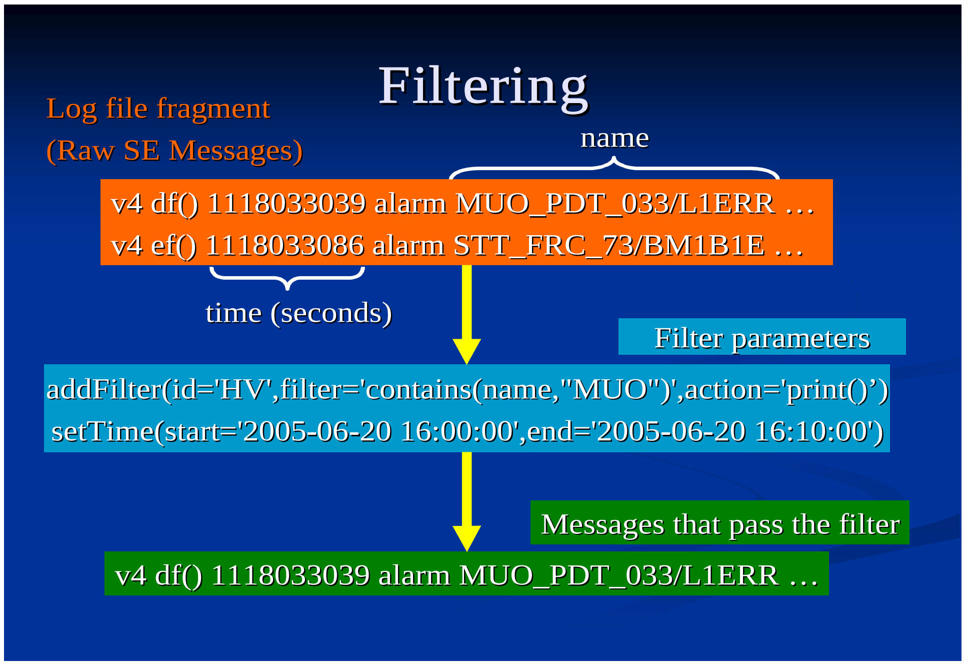
And being that the action stated on the configuration file says print, the output would look somewhat like this:

```
File Set contains 1 files :
/online/log/ses/logs/se_log.20050620-000000CDT.gz
100000 lines were read in successfully, lap time is 6.59 secs
200000 lines were read in successfully, lap time is 13.04 secs
300000 lines were read in successfully, lap time is 20.04 secs
400000 lines were read in successfully, lap time is 26.75 secs
=================================================================
Action Node PRINT, filter HV :
2005-06-20  16:04:52  =>  v4  df()  1119301492  alarm  MUO_PDT_033/L1ERR
0d0olctl66 0 none none good major analog mbbi 7 L1 OK
2005-06-20  16:04:52  =>  v4  df()  1119301492  alarm  MUO_PDT_033/FEB  0
d0olctl66 0 none none good major analog mbbi 7 FEB NORMAL
2005-06-20  16:04:54  =>  v4  df()  1119301494  alarm  MUO_PDT_033/L1ERR  0
d0olctl66 0 none none bad major analog mbbi 7 L1 Error
```

If for instance, the action specified was “stat”, or “count”, or some other form of action, the message displayed would have a different format. If the date is not specified in the configuration file, then the command on the command line would look somewhat like this:

```
sesBrowser.py browser.fil ‘2005-06-20 16:00:00’ ‘2005-06-20 16:05:00’
```

The exact same results would be produced either way, as long as the filter parameters in the configuration file remain unchanged.



```
v4 df() 1118033039 alarm MUO_PDT_033/L1ERR …
v4 ef() 1118033086 alarm STT_FRC_73/BM1B1E …
```

```
addFilter(id='HV',filter='contains(name,"MUO")',action='print()’)
setTime(start='2005-06-20 16:00:00',end='2005-06-20 16:10:00')
```

```
v4 df() 1118033039 alarm MUO_PDT_033/L1ERR …
```

LIMITATIONS OF THE SESBROWSER.PY PROGRAM

Like every bright and wonderful innovation, there are always limitations, and these limitations are reduced by updating these technologies, to reduce the difficulty and increase the ease of using these programs.

First of all, to use the sesBrowser.py program, one has to be using a Linux, D-Zero computer, which is connected online to one of the D-Zero online groups. One has to also set up a configuration file that is going to be run by the program, and specify the dates required. Before running the program, one also has to enter a “setup d0online” command that just makes the appropriate software required for the execution of the program available. What happens if the person doesn’t know the exact format to enter the dates? Or the person doesn’t know the exact manner to format the configuration file, so the sesBrowser.py program can run it? What happens if the person is not in the D-Zero area and needs to work? Or even, what happens if the person needs to work form home?

REMOVING LIMITATIONS:

HOW SESBROWSER.HTML AND SESBROWSER.JAVA PROGRAMS WORK

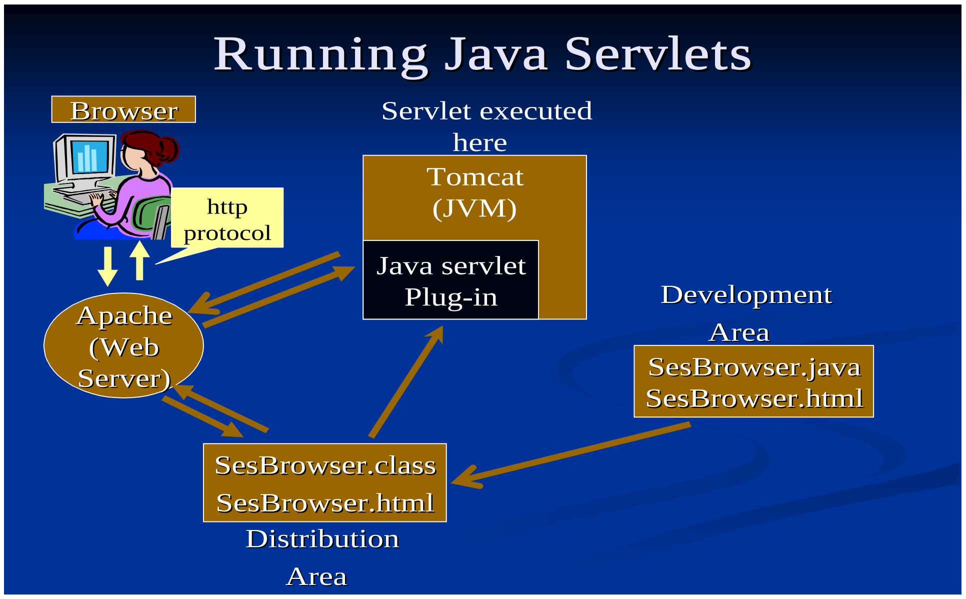
As earlier stated, the aim of any advancing technology is to reduce to difficulty level of an existing technology, or create an easier way of doing things that have been done in the past by traditional methods. With all the limitations stated above, my project comes in to put all these limitations in check and increase the level of convenience. These two programs work almost exactly like the python program already written, just that this time, it is made available over the web, making it accessible from anywhere in the world. They create a user friendly environment, which enables the user to specify the detectors, action, dates and everything requested. This eliminates the need for a configuration file, and also eliminates errors in passing parameters by picking everything required from drop down boxes on a web page. It would be redundant to create an entirely new program for this purpose, and it would also make the effort of creating the sesBrowser.py program wasted. Instead, a program has been written to run the sesBrowser.py program, by creating the configuration file, and initializing all the software needed for the execution of the sesBrowser.py program. Here is a list of software technologies used to make this project a reality:

- Java: This is a programming language developed by Sun Microsystems. It is characterized by the fact that programs written in Java do not rely on an operating system. This is achieved by the use of "virtual machines" that allow Java programs to run on a particular operating system. One of the common uses of Java is to create Applets that run inside a web page.
- Java Servlets: Servlets are java objects that extend the functionality of a web server. A servlet can read data from an HTTP request, and write data as an HTTP response. In this manner, it interacts with the browser. Applets and Servlets are generally used to build web applications.
- HTML: Hypertext Markup Language is a language used to create web pages and other documents that can contain text, graphics, and connections called hyperlinks.
- Tomcat: Functions as a servlet container developed under the Jakarta Project at the Apache Software Foundation. Tomcat implements the servlet and the Java Server Pages (JSP) specifications from Sun Microsystems. It's considered to be an application server.
- Apache: Apache is a Unix-based, open-source Web server that is used to host about half the sites on the Internet. Originally, Apache was a Unix product, but now versions for Windows, OS/2 and other platforms exist. As with most open-source projects, there are

numerous add-ons and tailored versions of the server available, which are created using the Apache module API. The name comes from its origins as a series of "patch files."

All the programs for this project are written in Java, and HTML. HTML is used to for the webpage where a user enters the parameters. The other programs are written in java with support for communicating with a web server.

INFRASTRUCTURE



The browser, which can be any internet browser, like Internet Explorer, or Netscape, passes a url to the server, which is Apache, via a HTTP request. In this case, Apache gets the required information (SesBrowser.html) and passes this request to Tomcat, which is a web server extension that is a java virtual machine. Tomcat has the ability to execute Java Servlets, but Apache can not, that’s why Apache sends the request Tomcat. Tomcat and Apache communicate via a special protocol, AJPv13, using the TCP/IP Socket. The source codes necessary are written and stored in the development area. Tomcat cannot access the development area, but it can access the distribution area. The appropriate files are made available to the distribution area, and Tomcat accesses it, executes the program, and returns the output, in html, back to Apache.

THE DEVELOPMENT AREA:

The development area as mentioned, is the area where all the source codes necessary are written. In this case, the name of the development area is called the ses_servlet directory. The ses_servlet directory currently contains two files, and four directories.

- Build.sh
- Build.xml
- The lib directory
- The etc directory
- The src directory
- And the web directory

Build.sh:

- It is found in the ses_servlet directory
- Its purpose is to find the class paths for all the classes necessary for the execution of the servlets, and execute the build.xml file, which is the file used to compile the source codes, and make them available to a directory where Tomcat can access them.

Build.xml:

- When Java programs are compiled, two files are created. One with the '.java' extension, which is the source code, and the other with the '.class' extension, which is what the computer can understand. The build.xml file compiles all the java servlets from all the appropriate subdirectories in the ses_servlet directory. It creates the directory and all the subdirectories in the distribution area, making it available to Tomcat. When these programs are compiled, the source codes remain in the development area, but the '.class' files are trasfered to the distribution area. The source code remains in the development area so as to prevent access from users . The `build.xml` file provides several "targets" that support optional development activities (such as creating the associated Javadoc documentation, erasing the deployment home directory so you can build your project from scratch, or creating the web application archive file so you can distribute your application.

The src/ directory:

- This directory contains the Java source files that generate the servlets, beans, and other Java classes required by the application. If your source code is organized into packages (highly recommended for large projects), the package hierarchy should be reflected as a directory structure underneath this directory.

The etc/ directory:

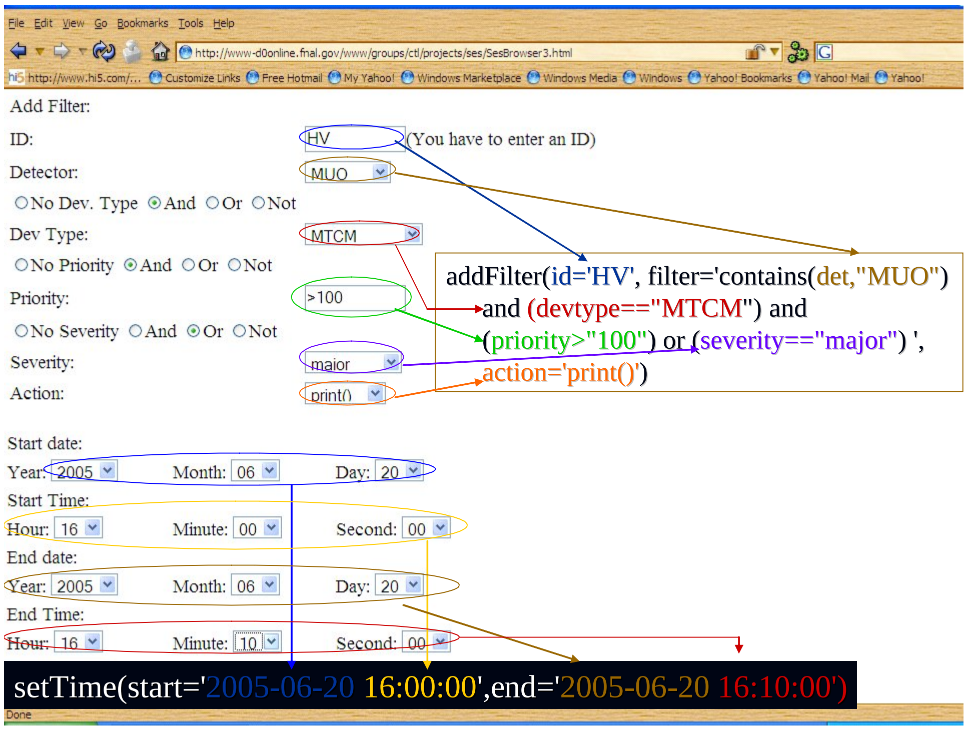
- This is the directory containing special files related to the application that will be copied to the Tomcat directory. In all cases, this will include the application deployment descriptor file (`web.xml`), but may include others as well.

The web/ directory:

- Directory containing the HTML files, JSP pages, and other resource files (such as JavaScript and stylesheet files) that will be accessible to browser clients. The entire hierarchy underneath this directory will be copied to the document root directory of your deployment home.

The lib directory:Directory containing JAR files that will be copied to the `WEB-INF/lib` deployment directory.

CURRENT SESBROWSER.HTM L PAGE



http://www-d0online.fnal.gov/www/groups/ctl/projects/ses/SesBrowser3.html

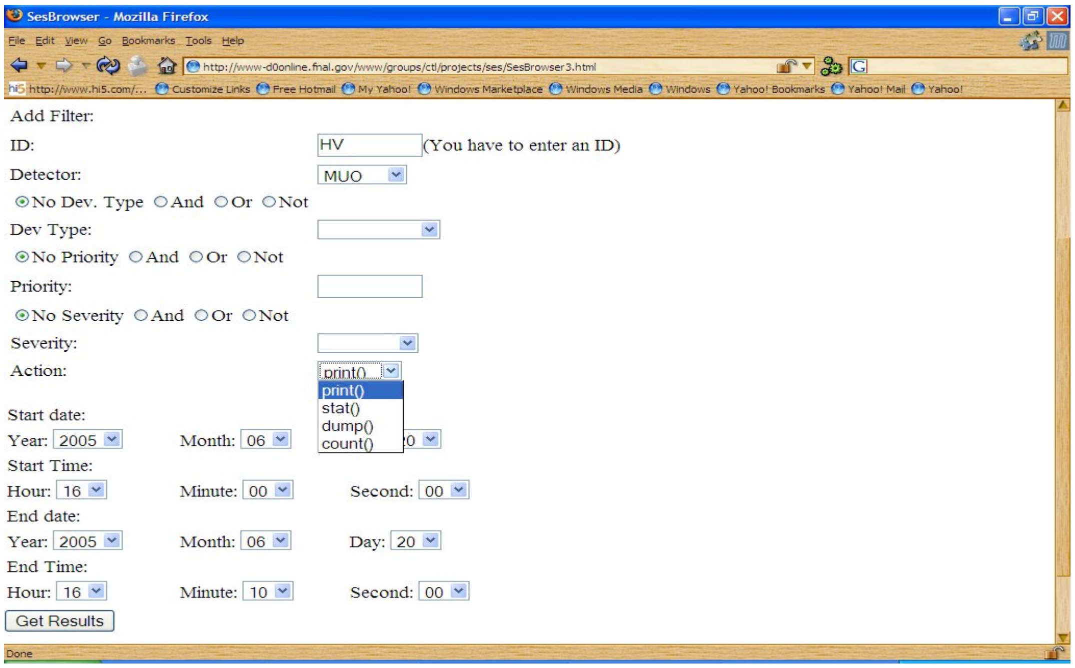
Add Filter:
ID: HV (You have to enter an ID)
Detector: MUO
○No Dev. Type [✓ And] ○Or ○Not
Dev Type: MTCM
○No Priority [✓ And] ○Or ○Not
Priority: >100
○No Severity ○And [✓ Or] ○Not
Severity: major
Action: print()

Start date:
Year: 2005 Month: 06 Day: 20
Start Time:
Hour: 16 Minute: 00 Second: 00
End date:
Year: 2005 Month: 06 Day: 20
End Time:
Hour: 16 Minute: 10 Second: 00

```
addFilter(id='HV', filter='contains(det,"MUO")
and (devtype=="MTCM") and
(priority>"100") or (severity=="major") ',
action='print()')
```

```
setTime(start='2005-06-20 16:00:00',end='2005-06-20 16:10:00')
```

The web page above shows the relationship between the current webpage, and the configuration file created. Below is the webpage with certain parameters specified, and the action being print.



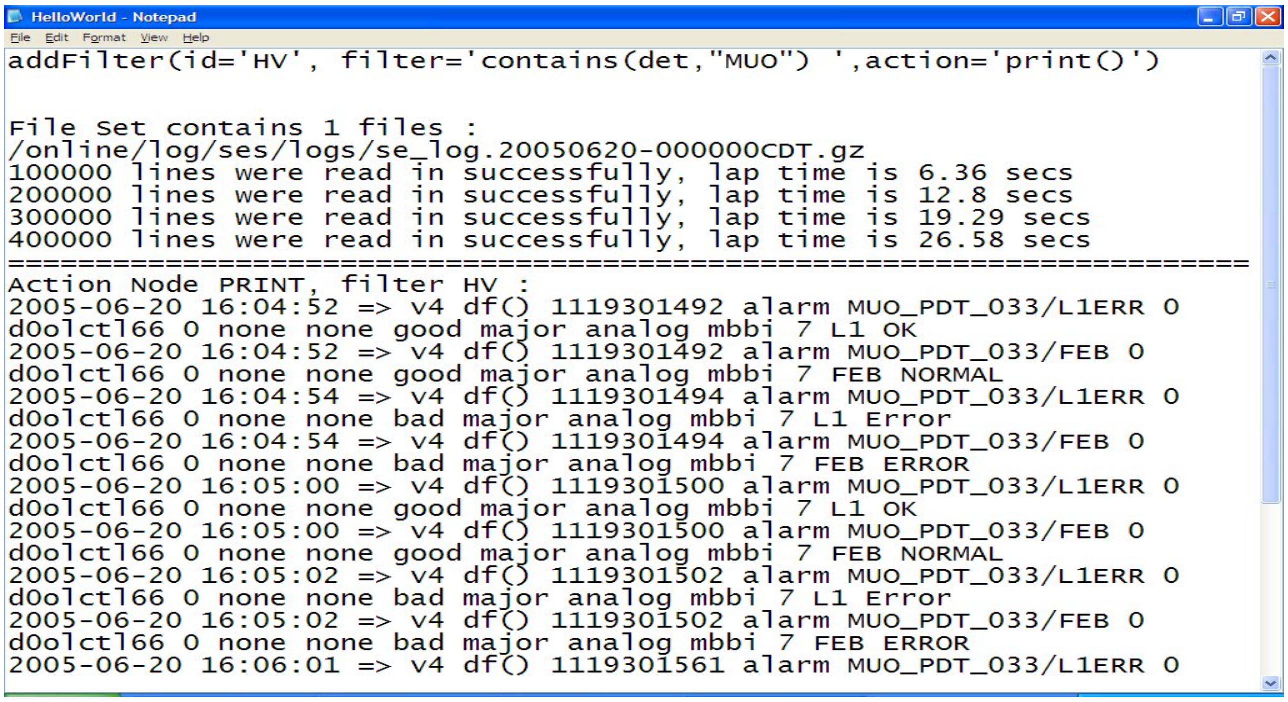
SesBrowser - Mozilla Firefox

http://www-d0online.fnal.gov/www/groups/ctl/projects/ses/SesBrowser3.html

Add Filter:
ID: HV (You have to enter an ID)
Detector: MUO
[✓ No Dev. Type] ○And ○Or ○Not
Dev Type:
[✓ No Priority] ○And ○Or ○Not
Priority:
[✓ No Severity] ○And ○Or ○Not
Severity:
Action: print()

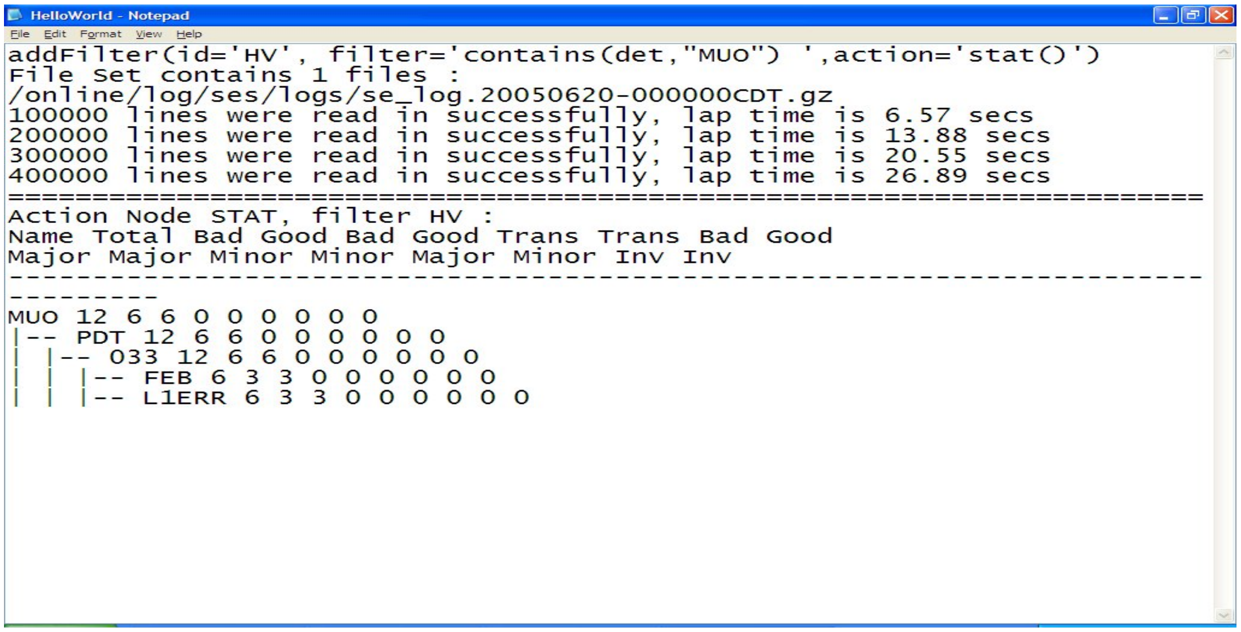
Start date:
Year: 2005 Month: 06
Start Time:
Hour: 16 Minute: 00 Second: 00
End date:
Year: 2005 Month: 06 Day: 20
End Time:
Hour: 16 Minute: 10 Second: 00

The results would look like:



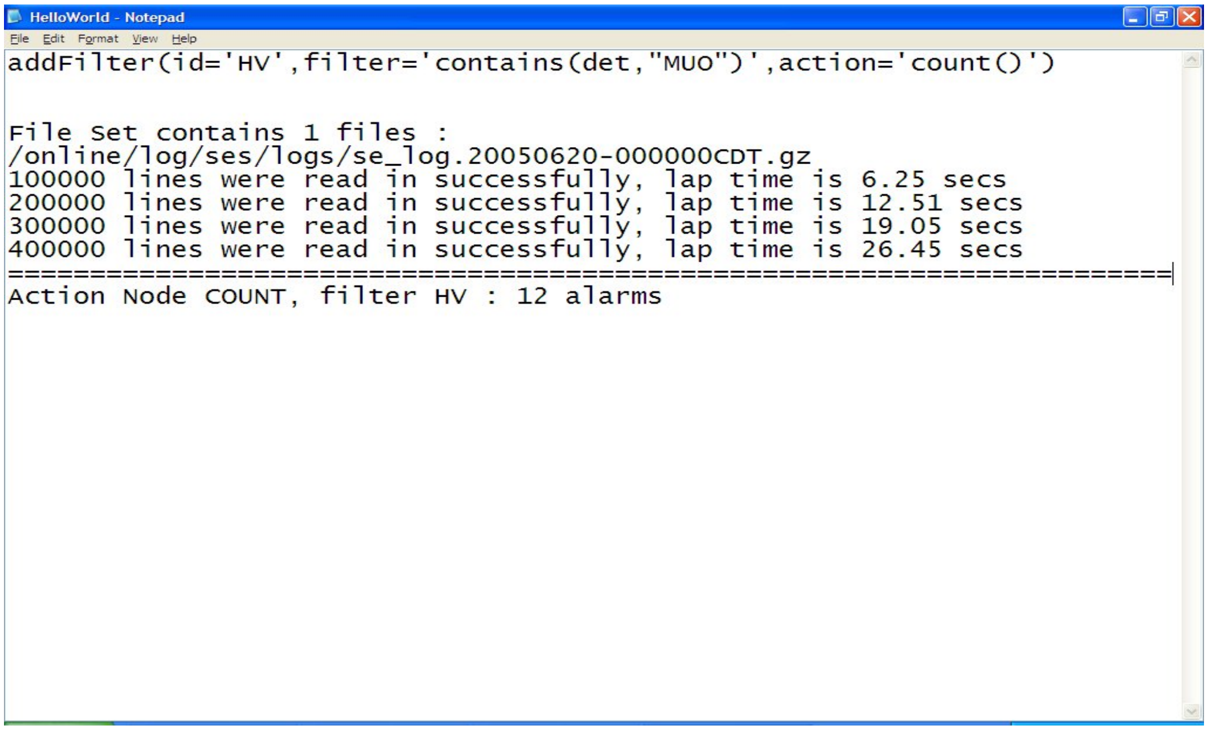
```
addFilter(id='HV', filter='contains(det,"MUO") ',action='print()')


File Set contains 1 files :
/online/log/ses/logs/se_log.20050620-000000CDT.gz
100000 lines were read in successfully, lap time is 6.36 secs
200000 lines were read in successfully, lap time is 12.8 secs
300000 lines were read in successfully, lap time is 19.29 secs
400000 lines were read in successfully, lap time is 26.58 secs
=========================================================================
Action Node PRINT, filter HV :
2005-06-20 16:04:52 => v4 df() 1119301492 alarm MUO_PDT_033/L1ERR 0
d0olctl66 0 none none good major analog mbbi 7 L1 OK
2005-06-20 16:04:52 => v4 df() 1119301492 alarm MUO_PDT_033/FEB 0
d0olctl66 0 none none good major analog mbbi 7 FEB NORMAL
2005-06-20 16:04:54 => v4 df() 1119301494 alarm MUO_PDT_033/L1ERR 0
d0olctl66 0 none none bad major analog mbbi 7 L1 Error
2005-06-20 16:04:54 => v4 df() 1119301494 alarm MUO_PDT_033/FEB 0
d0olctl66 0 none none bad major analog mbbi 7 FEB ERROR
2005-06-20 16:05:00 => v4 df() 1119301500 alarm MUO_PDT_033/L1ERR 0
d0olctl66 0 none none good major analog mbbi 7 L1 OK
2005-06-20 16:05:00 => v4 df() 1119301500 alarm MUO_PDT_033/FEB 0
d0olctl66 0 none none good major analog mbbi 7 FEB NORMAL
2005-06-20 16:05:02 => v4 df() 1119301502 alarm MUO_PDT_033/L1ERR 0
d0olctl66 0 none none bad major analog mbbi 7 L1 Error
2005-06-20 16:05:02 => v4 df() 1119301502 alarm MUO_PDT_033/FEB 0
d0olctl66 0 none none bad major analog mbbi 7 FEB ERROR
2005-06-20 16:06:01 => v4 df() 1119301561 alarm MUO_PDT_033/L1ERR 0
```

If the action specified is stat, the results would look like this:



```
addFilter(id='HV', filter='contains(det,"MUO") ',action='stat()')
File Set contains 1 files :
/online/log/ses/logs/se_log.20050620-000000CDT.gz
100000 lines were read in successfully, lap time is 6.57 secs
200000 lines were read in successfully, lap time is 13.88 secs
300000 lines were read in successfully, lap time is 20.55 secs
400000 lines were read in successfully, lap time is 26.89 secs
=======================================================================
Action Node STAT, filter HV :
Name Total Bad Good Bad Good Trans Trans Bad Good
Major Major Minor Minor Major Minor Inv Inv
-----------------------------------------------------------------------
---------
MUO 12 6 6 0 0 0 0 0 0
|-- PDT 12 6 6 0 0 0 0 0 0
| |-- 033 12 6 6 0 0 0 0 0 0
| | |-- FEB 6 3 3 0 0 0 0 0 0
| | |-- L1ERR 6 3 3 0 0 0 0 0 0
```

And finally, if the action specified is count, the results would look like this:



```
addFilter(id='HV',filter='contains(det,"MUO")',action='count()')


File Set contains 1 files :
/online/log/ses/logs/se_log.20050620-000000CDT.gz
100000 lines were read in successfully, lap time is 6.25 secs
200000 lines were read in successfully, lap time is 12.51 secs
300000 lines were read in successfully, lap time is 19.05 secs
400000 lines were read in successfully, lap time is 26.45 secs
=========================================================================
Action Node COUNT, filter HV : 12 alarms
```

FUTURE WORK

The SesBrowser.html, and SesBrowser.java has some limitations:

- Search time is very slow: Even with the fact that the search is available over the web, it is still very slow, because the java program does not actually conduct the search, but instead, it runs the python program, which by itself is very slow, so it takes the exact same time to run the java program, as it takes to run the python program (maybe a little longer, including the time to execute the java program).
- Inability to handle complex Filters: Manually configuring the configuration file, gives you the ability to add multiple filter lines, and handle complex requests. The SesBrowser

web system only allows you to enter one add filter line, which cannot handle complex filters. I would suggest that the html page be converted to java script, creating a dynamic web page that enables the users to add new filter tables on the click of a button. All the information can then be received, and then the required configuration file can be created. Another option would be to create another servlet that just creates a file, and writes all the information to it. Allowing you to write multiple filters, and on the submit button, the file already created would be read in, and the python program would run the already configured file.

CONCLUSION

This project is definitely a wonderful project because it actually advances directly the work done here at Fermilab. A lot of knowledge is required to put this project into reality. This project is running already, and the link to the web page is

http://www-d0online.fnal.gov/www/groups/ctl/projects/ses/SesBrowser.html

ACKNOWLEDGEMENTS

I would like to thank God first and foremost for giving me the wonderful opportunity to be here, he is the reason why I exist, and why I am so very proud of myself today. I would also like to thank my supervisor, Geoff Savage; words cannot express the amount gratitude for the time and patience he devoted to me throughout this summer. I really appreciate him. I would like to thank my fellow interns and staff here at DZero assembly building for helping me out with the little problems I had regarding working with a UNIX computer, and other little things that didn't seem like a lot, but in actuality, were the key to my success here. I wish to thank the SIST committee, Dr. Elliott McCrory, Mrs. Dianne Engram, Audrey Arns, Dr. Davenport and others for the

support I received from them individually. This program has given me the opportunity to realize dreams that I didn’t think would ever be a reality.

References:

Andrea Steelman and Joel Murach. Murach’s Java Servlets and JSP. Mike Murach and Associates, Inc.

Jason Hunter and William Crawford. Java Servlet Programming. O’Reilly & Associates, Inc., second edition, 2001.

Internet sources. www.pageresource.com, ww.w3schools.com